Sur quelques idées fausses ayant des conséquences en identification

Eric Walter — Michel Kieffer

Laboratoire des Signaux et Systèmes
CNRS – SUPELEC – Univ Paris-Sud
F-91192 Gif-sur-Yvette, France
Eric.Walter@lss.supelec.fr, Michel.Kieffer@lss.supelec.fr

RÉSUMÉ. *L'identification est plus un art qu'une science exacte, et fait appel autant à l'intuition qu'à des techniques très variées. La tradition est toujours tès présente, et il est important de remettre en question des coutumes dont les justifications peuvent parfois être essentiellement historiques. Citons, par exemple, la confiance accordée aux formules mathématiques classiques même quand elles se révèlent inadaptées à une mise en œuvre informatique, le rôle quasi exclusif des fonctions de coût quadratiques, l'utilisation d'approximations par différences finies pour les calculs de gradients, le recours quasi exclusif à des techniques locales aux résultats non garantis en identification et en simulation de modèles non linéaires. Pour tous ces aspects, nous donnons quelques pistes pour explorer d'autres voies.*

ABSTRACT. *Identification is more an art form than an exact science, and it is based on intuition just as much as on a large variety of techniques. Tradition is a major factor, and customs that may only have history as a justification should be questioned. Examples are the confidence bestowed on classical mathematical formulas even when they turn out not to be suited for computer implementation, the quasi-exclusive role played by the minimization of quadratic cost functions, the use of finite-difference approximations for the computation of gradients, the almost exclusive resort to local, non-guaranteed techniques for the identification and simulation of nonlinear models. For all of these topics, alternative approaches are suggested.*

MOTS-CLÉS : *calcul par intervalles, code adjoint, différentiation automatique, données aberrantes, estimation robuste, moindres carrés, optimisation globale.*

KEYWORDS: *adjoint code, automatic differentiation, global optimisation, interval analysis, least squares, outliers, robust estimation.*

RS - JESA – 42/2008. Identification des systèmes, pages 181 à 210

1. Introduction

L'identification est plus un art qu'une science exacte, et fait appel autant à l'intuition qu'à des techniques très variées. La tradition est toujours très présente, et il est important de remettre en question des coutumes dont les justifications sont parfois essentiellement historiques. Un premier exemple est la confiance accordée aux formules mathématiques classiques même quand elles ne sont pas adaptées à une mise en œuvre d'algorithmes sur ordinateurs. La section 2 illustre ce que l'on peut gagner par une reformulation numériquement robuste de l'algorithme des moindres carrés associée à un choix adapté des unités des variables et du protocole expérimental.

Un deuxième exemple est le rôle quasi exclusif joué en estimation depuis les travaux de Gauss et Legendre par la minimisation de fonctions de coût quadratiques, de type :

$$J(\mathbf{p}) = \sum_{i=1}^{n_\mathrm{t}} w(i) e^2(i, \mathbf{p}) \qquad [1]$$

où $\mathbf{p}$ est un vecteur de n_p paramètres à estimer, $w(i)$ est un coefficient de pondération positif à choisir, et $e(i, \mathbf{p})$ est une erreur. Cette dernière peut, par exemple, être définie comme :

$$e(i, \mathbf{p}) = y(i) - y_\mathrm{m}(i, \mathbf{p}) \qquad [2]$$

avec $y(i)$ une donnée recueillie sur le processus à modéliser et $y_\mathrm{m}(i, \mathbf{p})$ la sortie correspondante d'un modèle de paramètres $\mathbf{p}$. On parle alors d'*erreur de sortie* (associée à un modèle mis en parallèle sur le processus). D'autres types d'erreurs peuvent s'avérer plus adaptés, par exemple quand les sources d'imprécision majeures portent sur les entrées appliquées au processus et non pas sur la mesure de ses sorties.

Le choix [1] n'allait pas de soi. Dès 1632, Galilée utilise une fonction de coût en valeur absolue, de type :

$$J(\mathbf{p}) = \sum_{i=1}^{n_\mathrm{t}} w(i) \left| e(i, \mathbf{p}) \right| \qquad [3]$$

qui retient également l'attention de Boscovitch et Laplace (Farebrother, 1987). Laplace s'intéresse aussi dès 1786 à :

$$J(\mathbf{p}) = \max_{i=1}^{n_\mathrm{t}} w(i) \left| e(i, \mathbf{p}) \right| \qquad [4]$$

Or la fonction de coût [1] est aujourd'hui presque toujours utilisée. On peut y voir deux raisons principales. La première est que la valeur optimale $\widehat{\mathbf{p}}$ de $\mathbf{p}$ au sens de $J(\cdot)$ est donnée par une formule explicite simple, dans le cas important mais très particulier où $e(i, \mathbf{p})$ est affine en $\mathbf{p}$. C'était un avantage essentiel à l'époque des calculs à la main, mais qui ne peut plus guère servir de justification aujourd'hui... La seconde raison correspond à des hypothèses sur le caractère gaussien des erreurs, souvent contredites

par les faits. La section 3 parle des conséquences de cette violation des hypothèses et des outils que proposent les statistiques robustes pour y remédier.

Quand on ne dispose pas d'une expression explicite pour $\widehat{\mathbf{p}}$, on fait souvent appel à des algorithmes itératifs qui exploitent pour se déplacer dans l'espace des paramètres la valeur prise par le gradient de J au point courant à l'itération k :

$$\mathbf{g}(\widehat{\mathbf{p}}^k) = \frac{\partial J}{\partial \mathbf{p}}(\widehat{\mathbf{p}}^k) \qquad [5]$$

Comme $J(\widehat{\mathbf{p}}^k)$ est en général évalué par exécution d'un programme informatique qui peut être relativement complexe, la coutume, qui constitue notre troisième exemple, est d'évaluer $\mathbf{g}(\widehat{\mathbf{p}}^k)$ *via* une approximation par différences finies. La méthode du code adjoint, largement inspirée des travaux des automaticiens en commande optimale et que nous présentons en section 4, a pourtant deux avantages :

– c'est une *méthode exacte*, qui élimine l'erreur de méthode de l'approximation par différence finie ;

– elle est *remarquablement économe en calcul*, puisqu'il suffit pour évaluer $\mathbf{g}(\widehat{\mathbf{p}}^k)$ d'exécuter une seule fois un code déduit de celui servant à évaluer $J(\widehat{\mathbf{p}}^k)$ et dont la complexité est du même ordre de grandeur, et ceci quelle que soit la dimension de $\mathbf{p}$.

Un quatrième exemple est le recours quasi exclusif aux techniques itératives locales pour estimer des paramètres dès que l'on sort du cadre important mais très particulier de la minimisation d'une fonction de coût quadratique en $\mathbf{p}$. La section 5 présente des outils à base d'analyse par intervalles qui permettent d'éviter les limitations bien connues des approches locales et qui, contrairement aux techniques d'exploration aléatoire, sont à même de prouver leurs assertions.

Un dernier exemple est le sentiment qu'il n'est pas possible d'évaluer l'erreur de simulation globale de modèles formés de systèmes d'équations différentielles non linéaires et incertains. La section 6 montre que l'analyse par intervalles permet également l'intégration numérique garantie de tels systèmes.

2. L'élégance mathématique fait l'efficacité informatique

L'élégance et la concision d'un résultat mathématique ne se traduisent pas nécessairement par un algorithme efficace, et la méthode des moindres carrés en fournit un exemple éclairant. Soit un problème d'estimation de paramètres avec une fonction coût quadratique :

$$J(\mathbf{p}) = \mathbf{e}^{\mathrm{T}}(\mathbf{p})\mathbf{W}\mathbf{e}(\mathbf{p}) \qquad [6]$$

où $\mathbf{W}$ est une matrice symétrique définie positive connue. Supposons que l'erreur soit affine en $\mathbf{p}$ et puisse s'écrire :

$$\mathbf{e}(\mathbf{p}) = \mathbf{y} - \mathbf{F}\mathbf{p} \qquad [7]$$

avec $\mathbf{y}$ le vecteur de toutes les données expérimentales $y(i)$ et $\mathbf{F}$ une matrice tour (possédant plus de lignes que de colonnes). Par changement de variables il est facile de se ramener au cas où $\mathbf{W}$ est une matrice identité, et nous supposerons le changement de variables fait, de sorte que :

$$J(\mathbf{p}) = \mathbf{e}^{\mathrm{T}}(\mathbf{p})\mathbf{e}(\mathbf{p}) \qquad [8]$$

Si toutes les colonnes de $\mathbf{F}$ sont linéairement indépendantes, l'estimée optimale des paramètres est donnée par la formule classique :

$$\widehat{\mathbf{p}} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \qquad [9]$$

qu'il faut *soigneusement éviter d'utiliser* car elle se traduit par une augmentation drastique de la difficulté numérique du problème. Cette difficulté, ou plus précisément la sensibilité de la solution à une perturbation des données, est directement liée au *conditionnement* du système d'équations linéaires à résoudre pour calculer $\widehat{\mathbf{p}}$. Pour la norme spectrale, ce conditionnement est égal au rapport de la plus grande valeur singulière de $\mathbf{F}^{\mathrm{T}}\mathbf{F}$ à la plus petite. Il est donc supérieur ou égal à 1, et plus il est grand plus le problème est mal conditionné. Comme :

$$\mathrm{cond}(\mathbf{F}^{\mathrm{T}}\mathbf{F}) = (\mathrm{cond}(\mathbf{F}))^2 \qquad [10]$$

on a tout intérêt à éviter de passer par le calcul de $\mathbf{F}^{\mathrm{T}}\mathbf{F}$. Il suffit pour cela de passer par une *factorisation QR* de $\mathbf{F}$ (disponible dans toutes les bonnes bibliothèques de calcul matriciel) :

$$\mathbf{F} = \mathbf{QR} \qquad [11]$$

où $\mathbf{R}$ est une matrice triangulaire supérieure et $\mathbf{Q}$ une matrice de mêmes dimensions que $\mathbf{F}$ telle que $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$, et de résoudre :

$$\mathbf{R}\widehat{\mathbf{p}} = \mathbf{Q}^{\mathrm{T}}\mathbf{y} \qquad [12]$$

Cette solution, mathématiquement identique à celle donnée par [9], est informatiquement bien meilleure. Une autre approche, plus gourmande en calcul que la factorisation QR mais qui permet de traiter des cas particulièrement mal conditionnés, est de passer par une *décomposition en valeurs singulières* de $\mathbf{F}$ (elle aussi disponible dans toutes les bonnes bibliothèques de calcul matriciel) :

$$\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}} \qquad [13]$$

Dans [13], $\mathbf{U}$ est une matrice de mêmes dimensions que $\mathbf{F}$ telle que $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}$, $\boldsymbol{\Sigma}$ est une matrice diagonale dont les éléments diagonaux sont les valeurs singulières de $\mathbf{F}$ et $\mathbf{V}$ est une matrice carrée telle que $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$. Si toutes les colonnes de $\mathbf{F}$ sont linéairement indépendantes, la solution optimale est donnée, là aussi de façon mathématiquement équivalente à [9], par :

$$\widehat{\mathbf{p}} = \mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{U}^{\mathrm{T}}\mathbf{y} \qquad [14]$$

Si certaines colonnes de $\mathbf{F}$ sont linéairement dépendantes, la solution optimale au sens des moindres carrés n'est plus unique, et celle de norme euclidienne minimale est donnée par :

$$\widehat{\mathbf{p}} = \mathbf{V}\widetilde{\mathbf{\Sigma}}^{-1}\mathbf{U}^{\mathrm{T}}\mathbf{y} \qquad [15]$$

où $\widetilde{\mathbf{\Sigma}}$ est une matrice diagonale déduite de $\mathbf{\Sigma}$ en remplaçant tous les 0 situés sur la diagonale principale par des 1. Dans le cas plus réaliste où certaines colonnes de $\mathbf{F}$ sont *presque* linéairement dépendantes, le problème est mal conditionné, et une solution régularisée s'obtient en remplaçant par 0 dans $\mathbf{\Sigma}$ toutes les valeurs singulières inférieures à un seuil à choisir avant d'appliquer [15].

Exemple 1 *Cet exemple académique montre comment on peut améliorer le conditionnement numérique d'un problème d'identification non seulement en utilisant les bons algorithmes mais aussi en mettant les grandeurs à l'échelle et en optimisant le protocole de recueil des données. On souhaite fabriquer des brioches aussi hautes que possible en jouant sur quatre facteurs, qui forment un vecteur* $\mathbf{x}$ *:*

– x_1 *correspond à la vitesse d'incorporation des blancs dans la pâte, à choisir dans l'intervalle [100 g/mn, 200 g/mn],*

– x_2 *correspond à la durée de la cuisson, à choisir dans l'intervalle [40 mn, 50 mn],*

– x_3 *correspond à la température du four, à choisir dans l'intervalle [150 °C, 200 °C],*

– x_4 *correspond à la proportion de levure, à choisir dans l'intervalle [15 g/kg, 20 g/kg].*

Pour prédire la hauteur $y(\mathbf{x})$ *de la brioche on utilise le modèle polynomial :*

$$y_m(\mathbf{x}, \mathbf{p}) = p_1 + p_2x_1 + p_3x_2 + p_4x_3 + p_5x_4 + p_6x_2x_3 \qquad [16]$$

La sortie de ce modèle est linéaire en $\mathbf{p}$*, et la méthode des moindres carrés peut donc être appliquée pour trouver la valeur optimale des paramètres à partir d'un vecteur* $\mathbf{y}$ *de hauteurs de brioches constatées pour des valeurs variées de* $\mathbf{x}$*. Deux protocoles expérimentaux ont été suivis :*

– dans le premier, 160 brioches ont été fabriquées en tirant les valeurs de $\mathbf{x}$ *au hasard avec une loi uniforme sur le domaine autorisé,*

– dans le second, il y a eu également fabrication de 160 brioches, mais on a répété dix fois les seize expériences élémentaires différentes qu'il est possible d'obtenir quand on impose que les valeurs prises par les quatre facteurs soient toujours en limite haute ou basse de leur domaine autorisé. Ceci correspond à ce qu'on appelle un plan factoriel complet *à deux niveaux en théorie de la planification d'expériences, voir par exemple (Box* et al.*, 2005; NIST/SEMATECH, 2006).*

Pour chaque protocole, $\widehat{\mathbf{p}}$ *a été estimé en utilisant [9] et par l'approche par factorisation QR. Comme il s'agit d'un problème simple, la décomposition en valeurs*

	Cond **F**	Cond [9]	Cond QR
$\mathbf{x}$ aléatoire sans mise à l'échelle	1.4607×10^{6}	2.1336×10^{12}	1.4607×10^{6}
$\mathbf{x}$ aléatoire avec mise à l'échelle	2.8738	8.2587	2.8738
Plan factoriel avec mise à l'échelle	1	1	1

Tableau 1. *Evolution du conditionnement numérique*

singulières n'a été utilisée que pour calculer les conditionnements. Deux variantes ont été envisagées. Dans la première, les facteurs ont été utilisés tels quels, alors que dans la seconde chacun a subi une transformation affine de mise à l'échelle qui le ramène dans l'intervalle normalisé $[-1, 1]$. *Le tableau 1 résume le conditionnement obtenu dans chacun des cas.*

Sur un problème très simple, on constate ainsi que le conditionnement peut déjà se trouver multiplié par un facteur supérieur à 10^{12} *si l'on combine l'utilisation d'une formule mathématiquement exacte mais numériquement fragile avec un mauvais choix des unités et un protocole de recueil des données non optimisé.*

3. Les erreurs sont gaussiennes

La plupart de ceux qui se préoccupent de la nature statistique du bruit corrompant les données qu'ils vont utiliser pour identifier les paramètres de leur modèle font l'hypothèse d'un bruit blanc additif gaussien. Le mot gaussien est d'ailleurs souvent remplacé par *normal*, ce qui indique bien un état d'esprit que résume la boutade suivante, attribuée au physicien français Gabriel Lippmann (prix Nobel de physique en 1908 pour ses travaux sur la photographie couleur) :

Tout le monde croit que les erreurs sont normales, les mathématiciens parce qu'ils pensent que c'est un fait expérimentalement établi et les expérimentateurs parce qu'ils croient que c'est un théorème.

Le théorème *de la limite centrale* est souvent invoqué pour rassurer ceux qui viendraient à douter de ce caractère gaussien. Ce théorème dit que la somme de n variables aléatoires indépendantes et de même distribution (de moyenne μ et de variance σ^2 finie et non nulle) tend à être distribuée suivant une loi gaussienne de moyenne $n\mu$ et de variance $n\sigma^2$.

Une fois admis ce postulat sur la nature gaussienne des erreurs de mesure, la méthode du maximum de vraisemblance permet (sous certaines hypothèses supplémentaires) d'en déduire une procédure d'estimation de paramètres par minimisation d'une fonction de coût quadratique qui est quasi universellement utilisée.

Or l'hypothèse gaussienne est au mieux une approximation de la réalité. Des raisons multiples (caractère approximatif du modèle utilisé se traduisant par des erreurs déterministes, pannes momentanées de capteurs produisant des données aberrantes,

effet de non-linéarités...) font que cette hypothèse est souvent et clairement contredite par les faits. Comme le disait J.W. Tuckey en 1960 :

A tacit hope in ignoring deviations from ideal models was that they would not matter, that statistical procedures which were optimal under the strict model would still be approximately optimal under the approximate model. Unfortunately, it turned out that this hope was often drastically wrong ; even mild deviations often have much larger effects than were anticipated by most statisticians.

Il est ainsi facile de constater que la présence, par exemple, d'une faible proportion de données aberrantes suffit à entraîner des conséquences catastrophiques sur les paramètres estimés. Le caractère quadratique de la fonction de coût donne en effet un poids disproportionné à ces données atypiques. La prise de conscience de cette fragilité de l'inférence statistique classique a donné lieu au développement de méthodes statistiques robustes, avec en particulier les travaux de (Pearson *et al.*, 1936; Box *et al.*, 1975; Tuckey, 1960; Huber, 1981; Hampel, 1971; Hampel *et al.*, 1986; Rousseeuw, 1984; Rousseeuw *et al.*, 1987; Merrill *et al.*, 1971; Schweppe *et al.*, 1974; Mili *et al.*, 1990; Mili *et al.*, 1996).

Aujourd'hui encore, l'usage des statistiques robustes est loin d'être universellement répandu même chez les statisticiens. Elles ont en effet trois inconvénients principaux.

Le premier est qu'elles conduisent à diminuer l'influence de données particulièrement informatives, de sorte que l'efficacité des estimateurs robustes est moindre que celle des estimateurs non robustes, quand par chance les données vérifient les hypothèses sur lesquelles ces estimateurs non robustes sont fondés. D'où l'idée qui consiste à dire qu'il suffit de s'arranger pour collecter ses données proprement, afin de ne pas avoir à affronter les conséquences de la pollution de sa base de données par des données aberrantes. A ceci on peut répondre que ce sont parfois les données aberrantes qui constituent la partie la plus intéressante de la base, que l'élimination manuelle des données aberrantes est d'autant plus difficile que la base de données est grande (or les masses de données à traiter sont de taille sans cesse croissante) et enfin que si les erreurs sont distribuées suivant une loi à queue plus lourde que celle de la loi gaussienne (par exemple suivant une loi de Laplace) il sera impossible de détecter *a priori* celles qu'il convient d'éliminer. Nous avons donc besoin de méthodes qui ne s'écroulent pas dès que les données ne satisfont pas les hypothèses usuelles, quitte à comparer les résultats obtenus par ces méthodes à ceux fournis par les estimateurs classiques et à préférer ces derniers quand il n'y a pas contradiction. Notons que les tests statistiques classiques du caractère gaussien fonctionnent mal en présence de données aberrantes, de sorte qu'il est dangereux de s'en remettre à eux pour décider si un estimateur robuste doit être utilisé.

Le deuxième inconvénient est que l'optimisation des fonctions de coût liées à ces estimateurs robustes est souvent plus délicate que dans le cas d'un critère quadratique (non-différentiabilité, multimodalité, *etc.*). Il est, par exemple, sans espoir de tenter la minimisation d'une fonction de coût du type [3] (pourtant beaucoup plus robuste à des

données aberrantes qu'une fonction de type [1]) en invoquant un algorithme fondé sur un développement limité du coût (gradient, Newton, Gauss-Newton, quasi-Newton, gradients conjugués, *etc.*). Bien que la fonction de coût [3] soit dérivable presque partout, ces algorithmes se ruent sur les points où elle ne l'est pas pour y rester bloqués. L'existence d'algorithmes et de logiciels spécifiques rend cette objection beaucoup moins pertinente qu'autrefois.

Le dernier inconvénient est que le développement des méthodes d'estimation robuste n'est pas terminé, de sorte que le corpus des méthodes utilisables évolue et que certaines questions demeurent ouvertes.

Quoi qu'il en soit, les outils actuels permettent de donner des réponses au moins partielles à des questions aussi essentielles que (Hampel *et al.*, 1986) :

– les données sont-elles unanimes dans leur message ?

– si non, que dit la majorité, et où est la minorité ?

– quelles sont les données dont l'influence est telle qu'elles doivent bénéficier d'une attention particulière (concept de *point levier*) ?

– quel est le degré de protection offert contre les données aberrantes (concept de *point de rupture*) ?

– quelle est la perte d'efficacité subie par rapport à une méthode non robuste ?

– comment profiter des degrés de liberté dans les expériences réalisables pour augmenter la robustesse de l'estimation ?

Remarquons au passage que ce qui a été dit de la sensibilité des estimateurs classiques à des déviations par rapport à l'hypothèse d'erreurs gaussiennes est également vrai des déviations par rapport à l'hypothèse d'indépendance des erreurs, même si les techniques d'estimation robuste semblent moins développées dans ce domaine.

Exemple 2 *Considérons un ensemble de mesures* $y\left(i\right), i = 0 \ldots 20$, *que l'on cherche à décrire à l'aide du modèle* $y_m\left(i, \mathbf{p}\right) = p_1 + p_2 i$. *Un premier jeu de données sans bruit est généré par le modèle en prenant* $\mathbf{p}^* = \left(2, 1\right)^T$ *puis ces données sont corrompues par un bruit gaussien additif centré de variance unité. A partir des données bruitées résultantes, l'estimateur au sens des moindres carrés (MC) et celui minimisant la somme des valeurs absolues (VA) des erreurs fournissent dans ce premier cas des résultats proches :* $\widehat{\mathbf{p}}_{MC} = \left(1.87, 1.03\right)^T$ *et* $\widehat{\mathbf{p}}_{VA} = \left(2.13, 1.00\right)^T$, *voir la figure 1. Un second jeu de données est obtenu à partir du premier en forçant à zéro les mesures* $y\left(14\right)$ *à* $y\left(16\right)$ *pour simuler une panne de capteur. L'estimateur MC se révèle moins robuste à l'égard de ces données aberrantes que l'estimateur VA. En effet,* $\widehat{\mathbf{p}}_{MC} = \left(2.82, 0.68\right)^T$ *et* $\widehat{\mathbf{p}}_{VA} = \left(2.15, 0.99\right)^T$ *(voir la figure 2).*

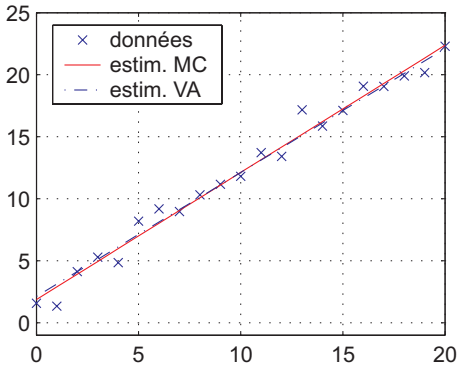

Figure 1. *Estimation de paramètres en l'absence de données aberrantes ; données et sorties des modèles obtenues par minimisation des fonctions de coût [1] et [3]*

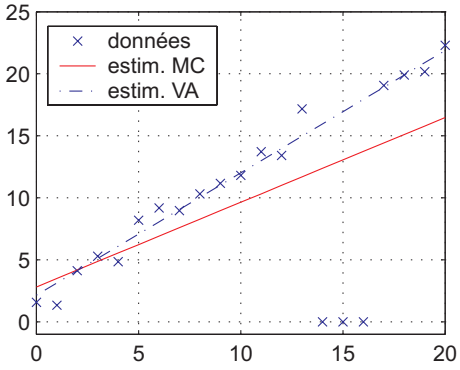

Figure 2. *Estimation de paramètres en présence de données aberrantes ; données et sorties des modèles obtenues par minimisation des fonctions de coût [1] et [3]*

Le calcul de l'estimateur VA passe par la minimisation d'une fonction de coût qui n'est pas différentiable partout, ce qui complique nettement l'opération. Certains *M-estimateurs* n'ont pas cet inconvénient. Un M-estimateur minimise :

$$J(\mathbf{p}) = \sum_{i=1}^{n_t} \rho\left(\frac{e(i,\mathbf{p})}{\sigma_i}\right) \qquad [17]$$

où, comme dans le cas de la méthode des moindres carrés pondérés, σ_i est un coefficient positif à choisir pour accorder plus ou moins de poids à la i^{e} erreur (quand toutes les erreurs ont le même poids, tous les σ_i sont pris égaux à 1). C'est la nature

de la fonction ρ qui définit le M-estimateur considéré (Huber, 1981; Rousseeuw *et al.*, 1987). Ainsi, l'*estimateur de Huber* utilise :

$$\rho(x) = \begin{cases} \frac{1}{2}x^2 & \text{si} \quad |x| \leqslant \delta \\ \delta|x| - \frac{1}{2}\delta^2 & \text{autrement} \end{cases} \qquad [18]$$

avec δ un seuil à choisir. Pour de petites valeurs de l'erreur, cet estimateur se comporte comme un estimateur MC, alors que pour les grandes erreurs il se comporte comme un estimateur VA. L'*estimateur de Tukey* utilise quant à lui :

$$\rho(x) = \begin{cases} \frac{1}{2}\left(x^2 - \frac{x^4}{\delta^2} + \frac{x^6}{3\delta^4}\right) & \text{si} \quad |x| \leqslant \delta \\ \frac{\delta^2}{6} & \text{autrement} \end{cases} \qquad [19]$$

ce qui donne un poids plus faible que les estimateurs VA et de Huber aux très grandes erreurs. Mentionnons enfin l'*estimateur M-Arctan* (Hassaine *et al.*, 2006) qui présente sur les estimateurs de Huber et Tukey l'avantage de toujours permettre une optimisation locale par la méthode de Gauss-Newton.

Les M-estimateurs assurent une protection efficace contre les données aberrantes quand toutes les données ont une importance à peu près analogue dans la détermination de $\widehat{\mathbf{p}}$. Ce n'est plus le cas quand il existe des *points leviers*, c'est-à-dire des données qui, à cause des conditions expérimentales qui leur sont associées, jouent un rôle critique. Quand on craint la présence de telles données, on peut faire appel à un M-estimateur généralisé (ou *GM-estimateur*) ; voir (Mili *et al.*, 1996) pour la mise en œuvre de cette idée due à Schweppe.

Les méthodes mentionnées jusqu'ici reposent implicitement sur l'hypothèse que les données aberrantes n'ont pas été organisées dans l'intention de pousser l'estimateur à la faute. On peut adopter une vision plus pessimiste, et imaginer qu'un mauvais génie a le pouvoir de remplacer α % des données expérimentales par des données aberrantes et de disposer ces dernières de façon à maximiser $\|\widehat{\mathbf{p}}_{\text{cor}} - \widehat{\mathbf{p}}_{\text{reg}}\|$, où $\widehat{\mathbf{p}}_{\text{cor}}$ est l'estimée obtenue à partir des données ainsi corrompues et $\widehat{\mathbf{p}}_{\text{reg}}$ l'estimée qui résulterait de l'emploi des données obtenues avant corruption. On appelle *point de rupture* d'un estimateur la plus petite valeur de α qui permet à cette norme de tendre vers l'infini. Deux estimateurs permettent d'assurer un point de rupture de presque $50\,\%$. Il s'agit de l'*estimateur de la moindre médiane du carré des résidus* (Rousseeuw, 1984) :

$$\widehat{\mathbf{p}} = \arg\min_{\mathbf{p}} \operatorname*{med}_i e^2(i, \mathbf{p}) \qquad [20]$$

qui ne tient aucun compte des presque $50\,\%$ des données qui sont associées aux résidus de plus grande valeur absolue, et de l'*estimateur des moindres carrés tronqués* :

$$\widehat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_{i \in \mathbb{I}} e^2(i, \mathbf{p}) \qquad [21]$$

où $\mathbb{I}$ est l'ensemble des indices i tels que $e^2(i, \mathbf{p})$ est inférieur ou égal à la médiane des carrés de tous les résidus, qui présente une meilleure efficacité asymptotique que l'estimateur [20]. Cette robustesse extraordinaire à des données aberrantes est acquise au

prix de la non-prise en compte de presque la moitié des données (que celles-ci soient aberrantes ou pas) et d'une optimisation qui requiert des algorithmes spécifiques.

Mentionnons enfin l'estimateur OMNE (*Outler Minimal Number Estimator*) (Lahanier *et al.*, 1987; Jaulin *et al.*, 2002), qui dans un contexte d'estimation à erreur bornée permet de se protéger contre des données aberrantes en proportion très supérieure à $50\ \%$, à condition toutefois que ces données aberrantes soient le fruit du hasard et non d'une malveillance.

4. Le calcul de gradient se fait par différences finies

De très nombreux algorithmes itératifs utilisés en identification exploitent la valeur prise par le gradient de la fonction de coût au point courant dans l'espace des paramètres. Cette évaluation représente alors une part importante de l'effort de calcul, et doit donc être simplifiée autant qu'il est possible. Sauf cas très particulier, on ne dispose pas d'une expression analytique de la fonction de coût, qui est souvent calculée par un programme complexe. La démarche standard consiste alors à utiliser une approximation par différences finies :

$$\frac{\partial J(\mathbf{p})}{\partial p_i} = \frac{1}{\Delta p_i}[J(\mathbf{p} + \Delta\mathbf{p}_i) - J(\mathbf{p})], i = 1 \dots n_\mathrm{p} \qquad [22]$$

où $\Delta\mathbf{p}_i$ est un vecteur dont toutes les composantes sont nulles sauf la i^e qui est égale à Δp_i. A supposer que Δp_i ait été bien choisi (ce qui n'a rien d'évident), ceci nécessite $n_\mathrm{p} + 1$ évaluations du coût et donc autant de simulations du modèle, ce qui peut se révéler très lourd si le modèle est complexe et le nombre de paramètres élevé. S'il faut procéder à un réglage de Δp_i la complexité des calculs augmente encore. Comme il est impossible de faire tendre Δp_i vers zéro à cause des problèmes numériques posés par le calcul de différences entre des nombres flottants très proches, le résultat est toujours approximatif. Une première approche évitant cet écueil consiste, quand c'est possible, à exprimer le gradient du critère en fonction des fonctions de sensibilité du premier ordre de la sortie du modèle, et à évaluer ces dernières. Considérons, par exemple, le modèle constitué par l'équation d'état :

$$\mathbf{x}' = \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, \mathbf{p}), \quad \mathbf{x}(0) = \mathbf{x}_0(\mathbf{p}) \qquad [23]$$

Supposons, pour simplifier, le vecteur des sorties du modèle linéaire en l'état :

$$\mathbf{y}_\mathrm{m}(t, \mathbf{p}) = \mathbf{C}(\mathbf{p})\mathbf{x}(t, \mathbf{p}) \qquad [24]$$

La sensibilité de $\mathbf{y}_\mathrm{m}$ par rapport au paramètre p_i est alors donnée par :

$$\frac{\partial \mathbf{y}_\mathrm{m}}{\partial p_i} = \frac{\partial \mathbf{C}}{\partial p_i}\mathbf{x} + \mathbf{C}\frac{\partial \mathbf{x}}{\partial p_i} \qquad [25]$$

et est donc facile à calculer à partir de la sensibilité de $\mathbf{x}$ par rapport à p_i, que nous noterons $\mathbf{s}_i$. En différentiant [23] par rapport à p_i, nous obtenons :

$$\frac{\mathrm{d}\mathbf{s}_i}{\mathrm{d}t} = \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{p})}{\partial \mathbf{x}^\mathrm{T}}\mathbf{s}_i + \frac{\partial \mathbf{f}(\mathbf{x}, \mathbf{p})}{\partial p_i}, \mathbf{s}_i(0) = \frac{\partial \mathbf{x}_0(\mathbf{p})}{\partial p_i}, i = 1 \dots n_\mathrm{p} \qquad [26]$$

Les n_{p} équations ainsi obtenues sont *indépendantes* (elles peuvent donc être simulées l'une après l'autre), *linéaires* (mais variables dans le temps à cause de la dépendance en l'état), et *leur partie homogène est identique*. Ces propriétés peuvent être exploitées pour simuler l'évolution de l'état et de ses n_{p} fonctions de sensibilité au premier ordre beaucoup plus vite qu'avec l'approche par différence finie (Valko *et al.*, 1984; Bilardello *et al.*, 1993), même si ceci demande (sauf dans le cas particulier où les équations d'état sont linéaires et les conditions initiales nulles) $n_{\mathrm{p}} + 1$ simulations, c'est-à-dire le même nombre qu'avec l'approche par différences finies.

L'*approche par code adjoint* (Speelpenning, 1980; Giering *et al.*, 1998) va permettre de dépasser cette limitation, et sera d'autant plus intéressante que la dimension de $\mathbf{p}$ est grande. Elle s'applique à tout coût évalué par un programme, pourvu que ce programme calcule ce coût d'une façon différentiable par rapport aux paramètres. Appelons *code direct* le programme qui évalue la valeur numérique du coût J pour une valeur numérique de ses variables d'entrée (vecteur de paramètres $\mathbf{p}$, vecteur des données expérimentales, *etc.*). Notons $\mathbf{v}$ le vecteur qui contient toutes les variables réelles du code, que celles-ci soient des variables d'entrée (les variables indépendantes) ou des résultats de calculs (les variables dépendantes). Ce vecteur peut être de très grande taille, et constitue seulement un intermédiaire de raisonnement qui ne recevra pas de matérialisation informatique. A un moment quelconque de l'exécution du code direct, la valeur de $\mathbf{v}$ peut être vue comme l'état de ce code, et l'exécution d'une instruction d'assignation (disons la k^{e}) se traduira par la modification d'une des composantes de ce vecteur d'état (disons la $\mu(k)^{\mathrm{e}}$), ce que l'on peut écrire :

$$v_{\mu(k)} = \varphi_k(\{v_i \mid i \in \mathbb{I}_k\}) \qquad [27]$$

où $\mathbb{I}_k$ est l'ensemble des indices des composantes de $\mathbf{v}$ qui sont des arguments de φ_k. Globalement, on peut donc voir le code direct comme un système décrit par une *équation d'état non linéaire à temps discret* :

$$\mathbf{v}(k) = \mathbf{\Phi}_k[\mathbf{v}(k-1)], k = 1 \ldots f \qquad [28]$$

où le rôle du temps est joué par le passage d'une instruction d'assignation à celle qui sera exécutée ensuite, et où :

$$[\mathbf{\Phi}_k(\mathbf{v})]_i = v_i, \forall i \neq \mu(k) \qquad [29]$$

$$[\mathbf{\Phi}_k(\mathbf{v})]_{\mu(k)} = \varphi_k(\{v_i \mid i \in \mathbb{I}_k\}) \qquad [30]$$

Notons que l'instruction associée à une valeur donnée de k dépend de l'ordre dans lequel les instructions sont exécutées. Les branchements demanderont donc une attention particulière. De façon arbitraire, et sans influence sur le résultat final, convenons de placer dans les premières composantes de $\mathbf{v}(0)$ les composantes de $\mathbf{p}$ puis celles des autres entrées du code direct. La valeur des autres composantes de $\mathbf{v}(0)$ n'a pas d'importance puis les variables dépendantes seront calculées par le code direct. Convenons

également qu'à la fin de l'exécution du code direct la valeur $J(\mathbf{p})$ du coût sera placée dans la dernière composante de l'état, de sorte que :

$$J(\mathbf{p}) = \left[\begin{array}{ccccc} 0 & \cdots & 0 & 1 \end{array} \right] \mathbf{v}(f) \qquad [31]$$

La règle des dérivations en chaîne permet d'écrire :

$$\frac{\partial J}{\partial \mathbf{p}} = \frac{\partial \mathbf{v}^{\mathrm{T}}(0)}{\partial \mathbf{p}} \frac{\partial \mathbf{\Phi}_1^{\mathrm{T}}}{\partial \mathbf{v}(0)} \cdots \frac{\partial \mathbf{\Phi}_{f-1}^{\mathrm{T}}}{\partial \mathbf{v}(f-2)} \frac{\partial \mathbf{\Phi}_f^{\mathrm{T}}}{\partial \mathbf{v}(f-1)} \frac{\partial J}{\partial \mathbf{v}(f)} \qquad [32]$$

On peut envisager d'effectuer les calculs correspondant à [32] de multiples façons, et en particulier de la gauche vers la droite ou de la droite vers la gauche. La façon la plus efficace en nombre d'opérations consiste à partir de la droite car on ne fait alors que des produits de matrices par des vecteurs. Posons :

$$\mathbf{d}(f) = \frac{\partial J}{\partial \mathbf{v}(f)} = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \end{array} \right] \qquad [33]$$

et calculons en nous déplaçant vers la gauche suivant la formule :

$$\mathbf{d}(k-1) = \frac{\partial \mathbf{\Phi}_k^{\mathrm{T}}}{\partial \mathbf{v}(k-1)} \mathbf{d}(k), k = f \ldots 1 \qquad [34]$$

que l'on peut interpréter comme une récurrence à temps rétrograde. Le vecteur $\mathbf{d}$ ainsi calculé est connu sous le nom d'*état adjoint*. Compte tenu de la façon dont la fonction $\mathbf{\Phi}_k$ a été construite, $\partial \mathbf{\Phi}_k^{\mathrm{T}} / \partial \mathbf{v}(k-1)$ est une matrice identité dont la $\mu(k)^{\mathrm{e}}$ colonne a été remplacée par le vecteur $\partial \varphi_k / \partial \mathbf{v}(k-1)$. Ceci entraîne que :

$$d_i(k-1) = d_i(k) + \frac{\partial \varphi_k}{\partial v_i(k-1)} d_{\mu(k)}(k) \text{ si } i \neq \mu(k) \qquad [35]$$

$$d_{\mu(k)}(k-1) = \frac{\partial \varphi_k}{\partial v_{\mu(k)}(k-1)} d_{\mu(k)} \qquad [36]$$

Partant de l'état adjoint final $\mathbf{d}(f)$, on peut ainsi par récurrence calculer l'état adjoint initial $\mathbf{d}(0)$. D'après [32] :

$$\frac{\partial J}{\partial \mathbf{p}} = \frac{\partial \mathbf{v}^{\mathrm{T}}(0)}{\partial \mathbf{p}} \mathbf{d}(0) \qquad [37]$$

Or, avec nos conventions :

$$\frac{\partial \mathbf{v}^{\mathrm{T}}(0)}{\partial \mathbf{p}} = \left[\begin{array}{cc} \mathbf{I}_{n_{\mathrm{p}}} & \mathbf{0} \end{array} \right] \qquad [38]$$

Ceci revient à dire que le gradient $\mathbf{g}(\mathbf{p})$ est contenu dans les n_p premières composantes de $\mathbf{d}(0)$. Les composantes suivantes de $\mathbf{d}(0)$ contiennent les dérivées partielles de la fonction de coût par rapport à toutes les autres entrées du code direct (les données, par exemple). Ces dérivées partielles sont ainsi obtenues en prime, sans aucun calcul complémentaire. Notons qu'il n'est pas nécessaire de stocker les valeurs successives de l'état adjoint, car seul $\mathbf{d}(0)$ sera utile. C'est pourquoi nous n'indexerons pas temporellement les composantes de $\mathbf{d}$.

La k^e instruction d'affectation du code direct :

$$v_{\mu(k)} = \varphi_k(\{v_i \mid i \in \mathbb{I}_k\}) \qquad [39]$$

se traduit par les instructions adjointes suivantes, dont l'ordre doit être respecté :

$$\textbf{for all } i \in \mathbb{I}_k, i \neq \mu(k), \textbf{ do } \{d_i = d_i + \frac{\partial \varphi_k}{\partial v_i} d_{\mu(k)}\} \qquad [40]$$

$$d_{\mu(k)} = \frac{\partial \varphi_k}{\partial v_{\mu(k)}} d_{\mu(k)} \qquad [41]$$

Supposons, par exemple, que la k^e instruction d'affectation du code direct soit :

$$J = J + [y(t) - y_\mathrm{m}(t)]^2 \qquad [42]$$

de sorte que :

$$\varphi_k[J, y(t), y_\mathrm{m}(t)] = J + [y(t) - y_\mathrm{m}(t)]^2 \qquad [43]$$

Notons $\mathrm{d}J$, $\mathrm{d}y(t)$ et $\mathrm{d}y_\mathrm{m}(t)$ les variables adjointes associées aux variables arguments de φ_k, soit J, $y(t)$ et $y_\mathrm{m}(t)$. En appliquant [40] et [41] à [43], on obtient les instructions adjointes :

$$\mathrm{d}y(t) = \mathrm{d}y(t) + \frac{\partial \varphi_k}{\partial y(t)} \mathrm{d}J \qquad [44]$$

$$\mathrm{d}y_\mathrm{m}(t) = \mathrm{d}y_\mathrm{m}(t) + \frac{\partial \varphi_k}{\partial y_\mathrm{m}(t)} \mathrm{d}J \qquad [45]$$

$$\mathrm{d}J = \mathrm{d}J \qquad [46]$$

Si on élimine [46], qui est bien sûr superflue, il reste compte tenu de [43] :

$$\mathrm{d}y(t) = \mathrm{d}y(t) + 2[y(t) - y_\mathrm{m}(t)]\mathrm{d}J \qquad [47]$$

$$\mathrm{d}y_\mathrm{m}(t) = \mathrm{d}y_\mathrm{m}(t) - 2[y(t) - y_\mathrm{m}(t)]\mathrm{d}J \qquad [48]$$

Les instructions de branchement doivent elles aussi être dualisées. Les boucles (do, for, while) du code direct se transforment dans le code adjoint en des boucles où les instructions adjointes sont exécutées dans l'ordre inverse de leurs contreparties du code direct. En présence de branchements conditionnels, il faudra mémoriser le chemin suivi lors de l'exécution du code direct pour être en mesure d'exécuter les instructions du code adjoint dans l'ordre approprié. C'est ainsi que la présence dans le code direct de l'instruction de branchement :

if (C) **then** {code A} **else** {code B} [49]

se traduira dans le code adjoint par l'instruction de branchement :

if (C) **then** {adjoint de A} **else** {adjoint de B} [50]

La construction du code adjoint peut être résumée ainsi :

– associer à chaque variable réelle du code direct une variable adjointe (choisir une convention permettant facilement de reconnaître qu'une variable est adjointe et à quelle variable du code direct elle correspond),

– initialiser toutes les variables adjointes à zéro, sauf celle associée à la valeur du coût, qui sera initialisée à un,

– dualiser les instructions du code direct dans l'ordre inverse de celui de leur exécution, ce qui suppose d'inverser les boucles et de tenir compte des branchements conditionnels.

Le code direct est exécuté en premier, en prenant soin de mémoriser les branches exécutées et toutes les valeurs numériques dont la connaissance sera nécessaire à l'exécution du code adjoint (c'est-à-dire seulement celles qui interviennent de façon non linéaire comme arguments d'instructions d'affectation du code direct). Le code direct produit la valeur du coût $J(\mathbf{p})$. Le code adjoint est alors exécuté et produit la valeur du gradient $\mathbf{g}(\mathbf{p})$ de ce coût (Les valeurs des composantes de $\mathbf{g}(\mathbf{p})$ sont données par les valeurs prises en fin d'exécution du code adjoint par les variables adjointes associées aux paramètres).

Il faut être très soigneux dans la préparation du code adjoint, car des erreurs d'apparence anodine peuvent avoir des résultats désastreux. Une bonne pratique (Talagran, 1991) est de :

– développer le code adjoint à partir du code direct (et pas à partir des équations que ce dernier met en œuvre),

– donner au code adjoint une structure lisible (chaque variable ou sous-programme du code direct doit avoir sa contrepartie dans le code adjoint),

– ne jamais modifier le code direct sans actualiser le code adjoint.

La propriété suivante peut permettre de détecter des erreurs de dualisation. Soit $\delta\mathbf{v}(k)$ l'état de l'équation [28] linéarisée au voisinage de sa trajectoire nominale. Le

produit scalaire de $\delta\mathbf{v}(k)$ avec l'état adjoint $\mathbf{d}(k)$ reste constant le long de cette trajectoire, de sorte que :

$$\mathbf{d}^{\mathrm{T}}(k+1)\delta\mathbf{v}(k+1) = \mathbf{d}^{\mathrm{T}}(k)\delta\mathbf{v}(k), k = 0 \ldots f-1 \qquad [51]$$

Considérons, par exemple (Walter *et al.*, 1994; Walter *et al.*, 1997), le code direct :

```
J = 0 ;
ym(0) = p2 ;
% boucle directe
for i = 1 to nt do {
  ym(i) = p1*ym(i-1) ;
  J = J + [y(i) - ym(i)]^2 ;
}.
```

Il évalue une fonction de coût qui dépend de deux paramètres et de données expérimentales $y(i),\ i = 1 \ldots n_{\mathrm{t}}$. La technique que nous venons de décrire génère, de façon parfaitement automatisable, le code adjoint suivant :

```
% initialisation des variables adjointes
dJ = 1 ;
for i = 1 to nt do {
  dy(i) = 0 ;
  dym(i) = 0 ;
}
dp1 = 0 ;
dp2 = 0 ;
% boucle retrograde
for i = nt down to 1 do {
  % dualisation de la seconde
  % instruction de la boucle directe
  dy(i) = dy(i) + 2*[y(i) - ym(i)]*dJ ;
  dym(i) = dym(i) - 2*[y(i) - ym(i)]*dJ ;
  dJ = dJ ;
  % dualisation de la premiere
  % instruction de la boucle directe
  dym(i-1) = dym(i-1) + p1*dym(i) ;
  dp1 = dp1 + ym(i-1)*dym(i) ;
  dym(i) = 0 ;
}
% dualisation de l'instruction
% qui precede la boucle directe
dp2 = dp2 + dym(0) ;
dym(0) = 0.
```

La valeur fournie pour le gradient, contenue dans `dp1` et `dp2`, est exacte (aux erreurs d'arrondi introduites par l'ordinateur près). Il n'y a donc pas d'erreur de méthode

(contrairement à ce qui se passait avec l'approche par différences finies). Le gradient est évalué en deux exécutions (une du code direct et une du code adjoint), et ceci quel que soit le nombre des paramètres (contrairement à ce qui se passait avec l'approche par fonctions de sensibilité). Il faut pour cela disposer du code direct, et de suffisamment de mémoire pour stocker les valeurs calculées par le code direct qui sont nécessaires à l'exécution du code adjoint.

La génération automatique de code adjoint est un point clé pour la dissémination de ce type de technique, surtout dans les domaines où le code direct est de grande taille et subit de fréquentes modifications (comme c'est le cas, par exemple, en climatologie). On trouvera dans (Naumann *et al.*, 2006) la description d'un tel générateur automatique et de son application à une configuration simplifiée du modèle de circulation générale du MIT (environ 40 000 lignes de code, pour l'essentiel en Fortran 77). Ce modèle comporte déjà 10^5 variables à optimiser, mais l'objectif est de passer à 10^8 variables, ce qui suppose d'améliorer encore l'efficacité de la dualisation. Le site `http://www.autodiff.org/` regroupe de nombreuses informations utiles sur l'état de l'art.

5. L'estimation non linéaire passe par des techniques locales

L'estimée des paramètres est en général définie par référence à un critère :

$$\widehat{\mathbf{p}} = \arg\min_{\mathbf{p}\in\mathbb{P}} J(\mathbf{p}) \qquad [52]$$

où $\mathbb{P}$ est le domaine admissible *a priori* pour le vecteur des paramètres. Contrairement à ce que la notation précédente peut laisser à penser, rien ne garantit que $\widehat{\mathbf{p}}$ soit unique. On cherche donc l'ensemble des arguments du minimum global de $J(\cdot)$ sur $\mathbb{P}$, arguments que nous appellerons les *minimiseurs globaux* de $J(\cdot)$. Quand on ne dispose pas d'une expression analytique pour la valeur de $\widehat{\mathbf{p}}$, il est d'usage d'employer des techniques itératives locales. Soit $\widehat{\mathbf{p}}^k$ l'estimée à l'itération k ; partant d'une estimée initiale $\widehat{\mathbf{p}}^0$ ces techniques calculent une suite de meilleures valeurs, au sens où :

$$J(\widehat{\mathbf{p}}^{k+1}) < J(\widehat{\mathbf{p}}^k) \qquad [53]$$

Cette approche pose des problèmes bien connus :

– comment choisir $\widehat{\mathbf{p}}^0$?

– quand arrêter d'itérer ?

– comment ne pas se faire piéger par des minimiseurs locaux ?

– peut-il y avoir plusieurs minimiseurs globaux (plusieurs valeurs de $\widehat{\mathbf{p}}$ également optimales), et si oui comment les trouver tous ?

Si les techniques d'exploration aléatoire (algorithmes génétiques ou recuit simulé, par exemple) contribuent à répondre à ces questions, elles ne sauraient garantir les résultats qu'elles produisent en temps fini. Les techniques à base d'analyse par intervalles dont nous allons parler maintenant n'ont pas cet inconvénient et permettent

dans certains cas de fournir en temps fini des résultats garantis, y compris pour des problèmes impliquant des modèles décrits par des équations différentielles non linéaires pour lesquelles on ne dispose pas de solution analytique. La présence d'incertitude sur le modèle peut également être prise en compte.

5.1. *Quelques notions de calcul par intervalles*

Un intervalle réel $[x] = [\underline{x}, \overline{x}]$ est un ensemble continu de nombres réels, contenant une infinité de nombres non représentables par un ordinateur. L'intervalle $[x]$ peut malgré tout être parfaitement représenté, dès lors que ses bornes $\underline{x}$ et $\overline{x}$ le sont. Il est alors possible d'étendre l'ensemble des opérations arithmétiques $\{+, -, \times, /\}$ aux intervalles de la manière suivante :

$$\forall \circ \in \{+, -, \times, /\}\,,\ [x] \circ [y] = \{x \circ y \mid x \in [x]\,, y \in [y]\} \qquad [54]$$

Ainsi :

$$[x] + [y] = \left[\underline{x} + \underline{y}, \overline{x} + \overline{y}\right] \qquad [55]$$

$$[x] - [y] = \left[\underline{x} - \overline{y}, \overline{x} - \underline{y}\right] \qquad [56]$$

$$[x] \times [y] = \left[\min\left(\underline{x}\underline{y}, \overline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\overline{y}\right),\ \max\left(\underline{x}\underline{y}, \overline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\overline{y}\right)\right] \qquad [57]$$

$$[x] / [y] = [x] \times \left[1/\overline{y}, 1/\underline{y}\right],\ \text{lorsque } 0 \notin [y] \qquad [58]$$

Si les bornes de l'intervalle résultat ne sont pas représentables par un ordinateur, le nombre représentable immédiatement inférieur ou supérieur est utilisé, suivant que l'on cherche à calculer la borne inférieure ou supérieure. Cette possibilité d'*approximation extérieure* est mise en œuvre grâce au contrôle de la direction d'arrondi proposé par la norme IEEE 754 (IEEE Computer Society, 1985). Ainsi, l'intervalle résultat fourni par un ordinateur *contiendra toujours* l'intervalle qui aurait été obtenu avec une machine disposant d'une précision infinie. Ceci permet de manipuler des intervalles sur un ordinateur et de *démontrer numériquement* certaines propriétés de fonctions sur des intervalles, comme illustré par l'exemple suivant.

Exemple 3 *Considérons la fonction $f(x) = x^4 - 4x^2$, à minimiser sur l'intervalle $[-10, 10]$. Supposons que f ait été évaluée en $x = 1$ et que l'ensemble des valeurs prises par f sur l'intervalle $[2, 4]$, soit disponible :*

$$f(1) = -3 \quad \textit{et} \quad f([2, 4]) = [0, 192] \qquad [59]$$

Ces résultats permettent de constater que $f(x) \geqslant 0$ pour tout $x \in [2, 4]$. Comme $f(1) = -3$, le minimum de f sur $[-10, 10]$ ne peut être supérieur à -3. L'intervalle $[2, 4]$ ne peut donc pas contenir de minimiseur global.

C'est le type d'argument présenté dans l'exemple 3 qui est mis en œuvre dans les algorithmes d'optimisation globale utilisant l'analyse par intervalles. La principale difficulté d'une démonstration numérique repose sur l'évaluation de l'ensemble des valeurs prises par une fonction sur un intervalle. Soit $f(\cdot)$ une fonction réelle définie sur un domaine $\mathcal{D} \subset \mathbb{R}$. L'ensemble des valeurs prises par cette fonction sur un intervalle $[x] \subset \mathcal{D}$ est donné par :

$$f([x]) = \{f(x) \mid x \in [x]\} \quad [60]$$

Pour des fonctions élémentaires monotones, cet ensemble est très simple à caractériser. Ainsi :

$$\exp([x]) = [\exp(\underline{x}), \exp(\overline{x})] \quad [61]$$

Par contre, les fonctions élémentaires non monotones, telles que l'élévation à une puissance positive paire ou le sinus, nécessitent la mise en œuvre d'algorithmes spécifiques pour obtenir l'ensemble exact des valeurs qu'elles prennent sur un intervalle quelconque. Ainsi, pour tout $k \in \mathbb{N}$:

$$[x]^{2k} = \begin{cases} \left[0, \max\left(\underline{x}^{2k}, \overline{x}^{2k}\right)\right] & \text{si } 0 \in [x] \\ \left[\min\left(\underline{x}^{2k}, \overline{x}^{2k}\right), \max\left(\underline{x}^{2k}, \overline{x}^{2k}\right)\right] & \text{sinon} \end{cases}$$

Pour plus de détails, voir par exemple (Hammer *et al.*, 1995; Jaulin *et al.*, 2001).

Pour une fonction $f(\cdot)$ quelconque, il n'est en général pas possible de décrire exactement l'ensemble $f([x])$ pour un intervalle $[x]$ quelconque. C'est pour contourner cette difficulté que les *fonctions d'inclusion* ont été introduites. Une fonction d'inclusion $[f](\cdot)$ associée à une fonction $f(\cdot)$ est une fonction à valeurs *intervalles*, qui pour tout $[x] \subset \mathcal{D}$ satisfait :

$$f([x]) \subset [f]([x]) \quad [62]$$

La fonction d'inclusion permet ainsi d'approximer un ensemble que l'on ne sait pas forcément calculer par un intervalle que l'on sait calculer et qui le contient. Lorsque dans [62] l'inclusion est une égalité, la fonction d'inclusion est dite *minimale*. Si ce n'est pas le cas, elle est dite *pessimiste*. Il existe une infinité de fonctions d'inclusion pour une fonction donnée. Trouver la fonction d'inclusion minimale n'est pas toujours facile, mais il est souvent possible de se contenter d'une fonction d'inclusion *convergente*, c'est-à-dire telle que $[f]([x])$ converge vers $f([x])$ lorsque la taille de l'intervalle $[x]$ tend vers zéro.

Plusieurs méthodes systématiques existent pour construire une fonction d'inclusion $[f](\cdot)$ associée à une fonction $f(\cdot)$ donnée. La plus simple consiste à remplacer toutes les occurences de la variable réelle x par la variable intervalle correspondante $[x]$. La fonction d'inclusion ainsi obtenue est appelée fonction d'inclusion *naturelle*.

Exemple 4 *Considérons à nouveau la fonction :*

$$f(x) = x^4 - 4x^2 \quad [63]$$

que l'on souhaite évaluer sur $[x] = [2, 4]$. *Une première fonction d'inclusion naturelle est :*

$$[f]_1([x]) = [x]^4 - 4[x]^2 \quad [64]$$

On a alors, $[f]_1([2, 4]) = [-48, 240]$, *qui contient bien l'ensemble des valeurs prises par* $f(\cdot)$ *sur* $[2, 4]$. *Le pessimisme de* $[f]_1(\cdot)$ *ne permet cependant pas de conclure directement que* $[2, 4]$ *ne contient pas de minimiseur global sur* $[-10, 10]$ *à partir de la connaissance de la valeur de* $f(1)$. *En utilisant une seconde fonction d'inclusion naturelle définie, après factorisation de* [63], *par :*

$$[f]_2([x]) = [x]^2\left([x]^2 - 4\right) \quad [65]$$

il est possible d'obtenir $[f]_2([2, 4]) = [0, 192]$, *qui correspond cette fois à l'ensemble des valeurs prises par* $f(\cdot)$ *sur* $[2, 4]$.

Bien souvent, une fonction d'inclusion naturelle n'est pas minimale, comme illustré par l'exemple 4. Ceci est lié au fait que toutes les occurences de la variable intervalle sont traitées comme si elles étaient indépendantes les unes des autres. Ainsi, diminuer le nombre des occurences des variables dans l'expression formelle de $f(\cdot)$ permet de réduire le pessimisme de la fonction d'inclusion obtenue. Une autre solution consiste à utiliser des fonctions d'inclusion plus élaborées, telles que les formes centrées ou les formes de Taylor (Hammer *et al.*, 1995; Jaulin *et al.*, 2001).

L'extension de l'arithmétique d'intervalles aux vecteurs et aux matrices et des fonctions d'inclusion au cas multivariable est immédiate. Pour indiquer qu'un vecteur ou qu'une matrice est de type intervalle, nous l'écrirons entre crochets. Un vecteur d'intervalles sera aussi appelé *pavé*.

5.2. *Optimisation globale déterministe*

Grâce à une fonction d'inclusion d'une fonction donnée, il est possible de prouver que cette dernière ne s'annule pas sur un intervalle ou qu'elle y reste strictement positive. Cette faculté est un élément clé de l'algorithme d'optimisation globale de Hansen (Hansen, 1992), dont une version simplifiée est présentée maintenant.

Soit $J : \mathbb{R}^n \to \mathbb{R}$, une fonction de coût deux fois continûment dérivable, dont on cherche l'ensemble $\mathcal{P}$ de *tous* les minimiseurs *globaux* appartenant à un pavé de recherche $[\mathbf{p}]_0 \subset \mathbb{R}^n$ donné. L'algorithme de Hansen va fournir une liste de pavés dont l'union contiendra $\mathcal{P}$ de manière garantie, à condition que le minimum global ne soit pas atteint sur la frontière de $[\mathbf{p}]_0$.

Schématiquement, l'algorithme de Hansen va couper $[\mathbf{p}]_0$ en pavés plus petits et les placer dans une liste de pavés $\mathcal{L}$. Tout pavé de $\mathcal{L}$ pour lequel il pourra démontrer qu'il ne contient pas de minimiseur global sera éliminé de $\mathcal{L}$. Trois tests d'élimination sont exploités : le *test du point milieu*, le *test de monotonicité* et le *test de convexité* :

1) supposons qu'un majorant $\widetilde{J}$ du minimum global de $J(\cdot)$ sur $[\mathbf{p}]_0$ soit disponible. Considérons un pavé $[\mathbf{p}]$ de $\mathcal{L}$ et posons $\left[\underline{j}, \overline{j}\right] = [J]([\mathbf{p}])$, où $[J](\cdot)$ est une fonction d'inclusion de $J(\cdot)$. Le *test du point milieu* vérifie si :

$$\underline{j} > \widetilde{J} \qquad [66]$$

auquel cas $[\mathbf{p}]$ ne peut contenir de minimiseur global (voir l'exemple 3). Il peut donc être éliminé de $\mathcal{L}$,

2) supposons qu'une fonction d'inclusion $[\mathbf{g}](\cdot)$ du gradient de $J(\cdot)$ soit disponible. Si pour un pavé $[\mathbf{p}]$ de $\mathcal{L}$, on a :

$$\mathbf{0} \notin [\mathbf{g}]([\mathbf{p}]) \qquad [67]$$

alors $J(\cdot)$ est strictement monotone sur $[\mathbf{p}]$, et le *test de monotonicité* ainsi défini permet d'éliminer $[\mathbf{p}]$ de $\mathcal{L}$,

3) si $\mathbf{p}$ est un minimiseur global non contraint de $J(\cdot)$, alors $J(\cdot)$ doit être convexe dans un voisinage de $\mathbf{p}$. Ceci implique que le Hessien $\mathbf{H}(\cdot)$ de $J(\cdot)$ soit défini non négatif dans un voisinage de $\mathbf{p}$. Une condition nécessaire en est que les éléments diagonaux de $\mathbf{H}(\cdot)$ soient positifs ou nuls. Supposons donc que des fonctions d'inclusion $[\mathbf{H}_{ii}](\cdot)$, $i = 1 \ldots n$, des éléments diagonaux de $\mathbf{H}(\cdot)$ soient disponibles. Si pour un pavé donné $[\mathbf{p}]$ de $\mathcal{L}$, on a :

$$\exists i \in \{1 \ldots n\} \text{ tel que } [\mathbf{H}_{ii}]([\mathbf{p}]) \subset \mathbb{R}^*_- \qquad [68]$$

alors $J(\cdot)$ ne peut être convexe sur $[\mathbf{p}]$, et $[\mathbf{p}]$ peut être éliminé de $\mathcal{L}$ par le *test de convexité* ainsi défini.

Pour mettre en œuvre le test du point milieu, à chaque itération de l'algorithme, une tentative d'amélioration du majorant $\widetilde{J}$ du minimum global de $J(\cdot)$ sur $[\mathbf{p}]_0$ est effectuée, par exemple par une simple évaluation de $J(\cdot)$ au centre de $[\mathbf{p}]$ ou à l'aide d'une technique d'optimisation locale. Les tests de monotonicité et de convexité nécessitent des fonctions d'inclusion du gradient et des éléments diagonaux du Hessien de $J(\cdot)$. Ces quantités peuvent être obtenues soit explicitement soit par différentiation automatique, voir (Rall, 1981), par exemple grâce à la technique du code adjoint (voir la section 4).

Les pavés de la liste $\mathcal{L}$ sont classés par valeurs croissantes de la borne inférieure de l'évaluation intervalle du critère. Ceci permet d'éliminer rapidement de larges portions de la liste $\mathcal{L}$ dès qu'une amélioration de $\widetilde{J}$ est obtenue. Lorsqu'un pavé $[\mathbf{p}]$ résiste aux trois tests d'élimination, il est coupé en deux, à condition que sa taille soit supérieure à un seuil ε_1 et que la taille de l'évaluation intervalle de la fonction de coût sur

$[\mathbf{p}]$ soit supérieure à un seuil ε_2. Les deux pavés résultants sont alors replacés dans $\mathcal{L}$. L'algorithme s'arrête lorsqu'aucun pavé de $\mathcal{L}$ ne peut plus être coupé. Les pavés restant dans $\mathcal{L}$ sont tous susceptibles de contenir un minimiseur global de $J(\cdot)$ sur $[\mathbf{p}]_0$.

La version intervalle de l'algorithme de Newton-Gauss-Seidel peut être utilisée pour réduire la taille des pavés $[\mathbf{p}]$ ayant résisté aux tests d'élimination. Cet algorithme peut également découper $[\mathbf{p}]$ en plusieurs parties de manière à isoler plusieurs minimiseurs globaux de $J(\cdot)$ inclus dans $[\mathbf{p}]$. Enfin, il permet, sous certaines conditions, de prouver l'*existence* et l'*unicité* d'un minimiseur global dans $[\mathbf{p}]$. Pour plus de détails, consulter (Hammer *et al.*, 1995). D'autres techniques d'accélération sont disponibles, voir par exemple (Neumaier, 2004).

Contrairement aux méthodes d'optimisation classiques, il n'est pas nécessaire de fournir une initialisation $\widehat{\mathbf{p}}^0$. Ici, seul un pavé de recherche $[\mathbf{p}]_0$ doit être fourni. Ce pavé peut être très grand. De plus, quelles que soient les valeurs des paramètres ε_1 et ε_2, *aucun* pavé contenant un minimiseur global n'a pu être éliminé. La taille des pavés obtenus permet d'évaluer la précision avec laquelle les minimiseurs globaux ont été localisés. Toutes ces informations, que ne peuvent pas fournir des techniques d'optimisation classiques, justifient l'accroissement du temps de calcul nécessaire à une optimisation globale garantie.

La section qui suit montrera que l'analyse par intervalles permet également de revenir sur une idée communément admise en ce qui concerne la simulation des modèles décrits par des équations différentielles ordinaires.

6. On ne peut rien dire sur l'erreur globale de simulation d'EDO non linéaires

L'évolution de très nombreux dispositifs (mécaniques, chimiques...) peut être décrite par des modèles constitués d'un système d'équations différentielles ordinaires du type :

$$\left\{ \begin{array}{l} \mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{p}, t) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{array} \right. \qquad [69]$$

où $\mathbf{x}(t) \in \mathbb{R}^n$ est l'état du modèle et $\mathbf{p} \in \mathbb{R}^{n_\mathbf{p}}$ est le vecteur de ses paramètres.

L'estimation de $\mathbf{p}$, par exemple à l'aide de techniques de minimisation d'une fonction de coût requiert le calcul des solutions de [69] pour différentes valeurs de $\mathbf{p}$. Lorsque l'on ne dispose pas de la solution explicite de [69], il faut faire appel à des techniques numériques, voir par exemple (Gear, 1971). La plupart ne permettent pas de caractériser de manière globale l'erreur commise lors de l'évaluation de $\mathbf{x}(t)$, qui peut être considérable. Le problème devient encore bien plus compliqué lorsque $\mathbf{x}(0)$ et $\mathbf{p}$ sont incertains et que l'on souhaite déterminer l'*ensemble* des solutions de [69] lorsque $\mathbf{x}(0) \in [\mathbf{x}_0]$ et $\mathbf{p} \in [\mathbf{p}_0]$. La solution classique consiste à discrétiser $[\mathbf{x}_0]$ et

$[\mathbf{p}_0]$ et à calculer un faisceau de solutions, sans garantie que l'enveloppe de ce faisceau contienne toutes les solutions possibles.

6.1. *Intégration numérique garantie*

Pour ce problème, le calcul par intervalles est à nouveau capable de fournir, à chaque instant, des pavés contenant de manière garantie l'ensemble des solutions de [69], avec $\mathbf{x}(0) \in [\mathbf{x}_0]$. Il est même possible de prendre en compte un vecteur de paramètres $\mathbf{p}$ appartenant à un pavé $[\mathbf{p}_0]$. La principale hypothèse est que $\mathbf{f}$ est $k-1$ fois continûment différentiable sur un ouvert $\mathcal{D}$ de $\mathbb{R}^n$. Ceci permet d'obtenir un développement de Taylor d'ordre k de la solution avec évaluation intervalle du reste. La résolution se fait de manière récursive et en deux étapes. Pour en simplifier la présentation, nous partirons de $t = 0$:

1) il s'agit dans un premier temps de prouver l'*existence* d'une solution entre $t = 0$ et $t = h$ et de calculer une *approximation extérieure* grossière de cette solution. Ainsi, il faut trouver $h > 0$ et $[\tilde{\mathbf{x}}_1] \subset \mathcal{D}$ tels que :

$$\forall \mathbf{x}(0) \in [\mathbf{x}_0] \text{ et } \forall t \in [0,h], \mathbf{x}(t) \in [\tilde{\mathbf{x}}_1]$$

2) à partir de $[\tilde{\mathbf{x}}_1]$, il faut ensuite calculer une approximation plus fine de l'ensemble des $\mathbf{x}(h)$.

Plusieurs solutions ont été proposées, et un certain nombre d'outils d'intégration numérique garantie sont disponibles. Le plus ancien est AWA (Lohner, 1987), écrit en Pascal. COSY (Berz *et al.*, 1998), développé en Fortran, et VNODE (Nedialkov *et al.*, 2001), écrit en C++, sont les outils les plus utilisés.

Ces techniques d'intégration se révèlent efficaces principalement lorsque $[\mathbf{x}_0]$ et $[\mathbf{p}]$ sont petits. Si ce n'est pas le cas, la taille des pavés obtenus a tendance à exploser, rendant de plus en plus difficile l'obtention d'une approximation extérieure grossière lors de la première étape de l'algorithme. Les théorèmes de (Müller, 1926) permettent de résoudre en partie le problème précédent (Walter *et al.*, 2006).

6.2. *Théorèmes de Müller*

L'idée est d'enfermer les solutions de [69] lorsque $\mathbf{x}_0 \in [\underline{\mathbf{x}}_0, \overline{\mathbf{x}}_0]$ et $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$ entre les solutions de systèmes dynamiques ne faisant plus intervenir de quantités incertaines. Seule une reformulation du théorème de Müller dans le cas général est donnée ici. Pour le cas particulier des systèmes coopératifs, voir par exemple (Kieffer *et al.*, 2003).

Théorème 1 (Existence) *Soit une fonction* $\mathbf{f}(\mathbf{x}, \mathbf{p}, t)$ *continue sur un domaine :*

$$\mathbb{T} : \begin{cases} \boldsymbol{\omega}(t) \leqslant \mathbf{x} \leqslant \boldsymbol{\Omega}(t) \\ \underline{\mathbf{p}}_0 \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}_0 \\ 0 \leqslant t \leqslant h \end{cases} \quad [70]$$

où $\omega_i(t)$ *et* $\Omega_i(t)$, $i = 1 \ldots n$, *sont continues sur* $[0, h]$ *et telles que :*

1) $\boldsymbol{\omega}(0) = \underline{\mathbf{x}}_0$ *et* $\boldsymbol{\Omega}(0) = \overline{\mathbf{x}}_0$

2) pour $i = 1 \ldots n$,

$$D^{\pm}\omega_i(t) \leqslant \min_{\underline{\mathbb{T}}_i(t)} f_i(\mathbf{x}, \mathbf{p}, t) \quad [71]$$

et

$$D^{\pm}\Omega_i(t) \geqslant \max_{\overline{\mathbb{T}}_i(t)} f_i(\mathbf{x}, \mathbf{p}, t) \quad [72]$$

où $D^{\pm}$ *représente la dérivation à droite et à gauche et* $\underline{\mathbb{T}}_i(t)$ *et* $\overline{\mathbb{T}}_i(t)$ *sont des sous-ensembles de* $\mathbb{T}$ *définis par :*

$$\underline{\mathbb{T}}_i(\tau) \quad : \quad \begin{cases} x_i = \omega_i(t) \\ \omega_j(t) \leqslant x_j \leqslant \Omega_j(t), \; j \neq i \\ \underline{\mathbf{p}}_0 \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}_0 \\ t = \tau \end{cases} \quad [73]$$

$$\overline{\mathbb{T}}_i(\tau) \quad : \quad \begin{cases} x_i = \Omega_i(t) \\ \omega_j(t) \leqslant x_j \leqslant \Omega_j(t), \; j \neq i \\ \underline{\mathbf{p}}_0 \leqslant \mathbf{p} \leqslant \overline{\mathbf{p}}_0 \\ t = \tau \end{cases} \quad [74]$$

Alors, pour tout $\mathbf{x}(0) \in [\underline{\mathbf{x}}_0, \overline{\mathbf{x}}_0]$ *et* $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$, *une solution de* $\mathbf{x}' = \mathbf{f}(\mathbf{x}, \mathbf{p}, t)$ *existe dans :*

$$\mathbb{E} : \begin{cases} 0 \leqslant t \leqslant h \\ \boldsymbol{\omega}(t) \leqslant \mathbf{x} \leqslant \boldsymbol{\Omega}(t) \end{cases} \quad [75]$$

et est égale à $\mathbf{x}(0)$ *à* $t = 0$.

Théorème 2 (Unicité) *De plus, si pour tout* $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$,

$\mathbf{f}(\mathbf{x}, \mathbf{p}, t)$ *est Lipschitzienne par rapport à* $\mathbf{x}$ *sur* $\mathcal{D}$

alors pour tout $\mathbf{x}(0) \in [\underline{\mathbf{x}}_0, \overline{\mathbf{x}}_0]$ *et* $\mathbf{p} \in [\underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0]$, *la solution précédente est unique.*

Pour obtenir $\boldsymbol{\omega}(t)$ et $\boldsymbol{\Omega}(t)$, il faut construire un système couplé :

$$\begin{cases} \underline{\mathbf{x}}' = \underline{\mathbf{g}}(\underline{\mathbf{x}}, \overline{\mathbf{x}}, \underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0, t), \underline{\mathbf{x}}(0) = \underline{\mathbf{x}}_0 \\ \overline{\mathbf{x}}' = \overline{\mathbf{g}}(\underline{\mathbf{x}}, \overline{\mathbf{x}}, \underline{\mathbf{p}}_0, \overline{\mathbf{p}}_0, t), \overline{\mathbf{x}}(0) = \overline{\mathbf{x}}_0 \end{cases} \quad [76]$$

dont la solution $\left(\boldsymbol{\omega}^{\mathrm{T}}(t), \boldsymbol{\Omega}^{\mathrm{T}}(t)\right)^{\mathrm{T}}$ satisfait les conditions du théorème 1.

6.3. *Estimation des paramètres d'un modèle de compartiments*

Les contributions de l'analyse par intervalles et de l'intégration garantie à l'estimation des paramètres d'un modèle à temps continu vont être illustrées sur un modèle de compartiments (Jacquez, 1972). Ce type de modèles, rencontré fréquemment en pharmacocinétique ou en chimie par exemple, est constitué d'un ensemble de réservoirs homogènes échangeant de la matière entre eux et avec le milieu extérieur.

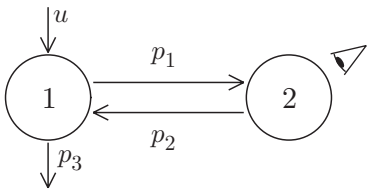

Figure 3. *Exemple de modèle à deux compartiments*

Le modèle à deux compartiments représenté sur la figure 3 peut être décrit par l'équation d'état :

$$\begin{cases} x_1' = -\left(p_1 + p_3\right) x_1 + p_2 x_2 \\ x_2' = p_1 x_1 - p_2 x_2 \end{cases} \quad [77]$$

où x_1 et x_2 représentent les quantités de matière contenue dans les compartiments 1 et 2. Le vecteur des paramètres à déterminer est $\mathbf{p} = (p_1, p_2, p_3)^{\mathrm{T}}$. Il correspond aux constantes cinétiques (positives) d'échange de matière entre les compartiments et avec l'extérieur. Une injection unité de matière est effectuée dans le compartiment 1 à $t_0 = 0$, ainsi $\mathbf{x}(0) = (1, 0)^{\mathrm{T}}$. Des mesures $y\left(t_i\right)$ de la quantité de matière dans le compartiment 2 sont effectuées aux instants t_i, $i = 1 \ldots N$, ainsi $y\left(t_i\right) = x_2\left(t_i\right)$. L'estimée de $\mathbf{p}$ au sens des moindres carrés minimise :

$$J\left(\mathbf{p}\right) = \sum_{i=1}^{n_{\mathrm{t}}} \left(y\left(t_i\right) - x_2\left(\mathbf{p}, t_i\right)\right)^2 \quad [78]$$

Nous avons vu dans la section 5.2 que le gradient et les éléments diagonaux du hessien permettent de prouver que certains pavés ne contiennent pas de minimiseur global de cette fonction de coût. Une solution pour obtenir le gradient voire les éléments diagonaux du hessien passe par l'évaluation de fonctions de sensibilité (voir la section 4). Notons s_{jk} la fonction de sensibilité du premier ordre de x_j par rapport à p_k :

$$s_{jk}\left(\mathbf{p}, t\right) = \frac{\partial x_j}{\partial p_k}\left(\mathbf{p}, t\right) \quad [79]$$

Le gradient de J peut alors être évalué comme :

$$\frac{\mathrm{d}J}{\mathrm{d}\mathbf{p}}(\mathbf{p}) = \begin{pmatrix} 2\sum_{i=1}^{n_t}\left(x_2(\mathbf{p},t_i) - y(t_i)\right)s_{21}(\mathbf{p},t_i) \\ 2\sum_{i=1}^{n_t}\left(x_2(\mathbf{p},t_i) - y(t_i)\right)s_{22}(\mathbf{p},t_i) \\ 2\sum_{i=1}^{n_t}\left(x_2(\mathbf{p},t_i) - y(t_i)\right)s_{23}(\mathbf{p},t_i) \end{pmatrix} \quad [80]$$

Les fonctions de sensibilité par rapport à un paramètre donné p_k satisfont un système d'équations différentielles obtenu en différentiant [77] par rapport à p_k. Il vient :

$$\begin{cases} s'_{11} = -(p_1+p_3)s_{11} + p_2 s_{21} - x_1 \\ s'_{21} = p_1 s_{11} - p_2 s_{21} + x_1 \\ s'_{12} = -(p_1+p_3)s_{12} + p_2 s_{22} + x_2 \\ s'_{22} = p_1 s_{12} - p_2 s_{22} - x_2 \\ s'_{13} = -(p_1+p_3)s_{13} + p_2 s_{23} - x_1 \\ s'_{23} = p_1 s_{13} - p_2 s_{23} \end{cases} \quad [81]$$

Comme $\mathbf{x}(0)$ ne dépend pas de $\mathbf{p}$, les conditions initiales pour chaque fonction de sensibilité sont nulles. Le système de huit équations différentielles ordinaires [77, 81] permet ainsi de calculer l'évolution de l'état et des fonctions de sensibilité du premier ordre. L'état initial est ici supposé connu. Par contre, le vecteur des paramètres est contenu dans un pavé. Le théorème 1 est donc mis en œuvre pour trouver des systèmes dynamiques déterministes dont les solutions permettent d'enfermer celles du système dynamique incertain.

Comme les p_k et les x_j sont tous positifs, l'obtention de ces systèmes dynamiques bornants est relativement simple. Ainsi, l'obtention de bornes inférieures et supérieures de $\mathbf{x}$, s_{11} et s_{21} peut passer par la résolution des deux systèmes différentiels :

$$\begin{cases} \underline{x}'_1 = -(\overline{p}_1+\overline{p}_3)\underline{x}_1 + \underline{p}_2\underline{x}_2 \\ \underline{x}'_2 = \underline{p}_1\underline{x}_1 - \overline{p}_2\underline{x}_2 \\ \overline{x}'_1 = -(\underline{p}_1+\underline{p}_3)\overline{x}_1 + \overline{p}_2\overline{x}_2 \\ \overline{x}'_2 = \overline{p}_1\overline{x}_1 - \underline{p}_2\overline{x}_2 \\ \underline{s}'_{11} = -(\overline{p}_1+\overline{p}_3)\underline{s}_{11} + \underline{p}_2\underline{s}_{21} - \overline{x}_1 \\ \underline{s}'_{21} = \underline{p}_1\underline{s}_{11} - \overline{p}_2\underline{s}_{21} + \underline{x}_1 \end{cases} \quad [82]$$

et

$$\begin{cases} \underline{x}'_1 = -(\overline{p}_1+\overline{p}_3)\underline{x}_1 + \underline{p}_2\underline{x}_2 \\ \underline{x}'_2 = \underline{p}_1\underline{x}_1 - \overline{p}_2\underline{x}_2 \\ \overline{x}'_1 = -(\underline{p}_1+\underline{p}_3)\overline{x}_1 + \overline{p}_2\overline{x}_2 \\ \overline{x}'_2 = \overline{p}_1\overline{x}_1 - \underline{p}_2\overline{x}_2 \\ \overline{s}'_{11} = -(\underline{p}_1+\underline{p}_3)\overline{s}_{11} + \overline{p}_2\overline{s}_{21} - \underline{x}_1 \\ \overline{s}'_{21} = \overline{p}_1\overline{s}_{11} - \underline{p}_2\overline{s}_{21} + \overline{x}_1 \end{cases} \quad [83]$$

Une procédure analogue peut être appliquée pour les fonctions de sensibilité du premier ordre par rapport à p_2 et à p_3. On obtient alors six systèmes de six équations différentielles ordinaires. Comme ces systèmes contiennent des parties communes, ils pourraient être regroupés en un seul système différentiel de seize équations seulement. Cependant, comme la complexité des outils d'intégration numérique garantie est proportionnelle au cube du nombre des équations différentielles du système à résoudre, il est préférable d'en rester aux systèmes différentiels initiaux.

Une approche similaire pourrait être envisagée pour l'obtention des fonctions de sensibilité du second ordre qui permettent d'évaluer le hessien du critère.

Pour la partie expérimentale, un modèle à deux compartiments décrit par [77] avec $\mathbf{p}^* = (0.6, 0.15, 0.35)^{\mathrm{T}}$ a été simulé et des données ont été collectées en arrondissant $x_2(t_i)$ à deux décimales pour $t_i = i$, avec $i = 1 \ldots 15$.

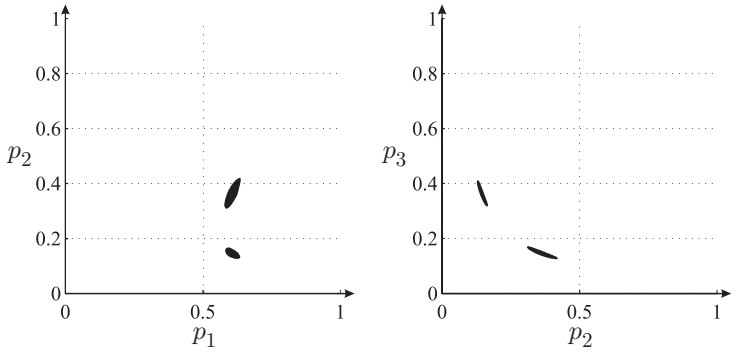

Figure 4. *Projection des ensembles solution obtenus*

Le domaine de recherche initial choisi est $[\mathbf{p}]_0 = [0.01, 1]^{\times 3}$. Pour des seuils $\varepsilon_1 = \varepsilon_2 = 0.001$, une version très simplifiée de l'algorithme d'optimisation globale décrit à la section 5.2, ne mettant en œuvre ni le test de convexité ni la version intervalle de l'algorithme de Newton-Gauss-Seidel, fournit une liste de pavés dont les projections sur les plans (p_1, p_2) et (p_2, p_3) sont représentées sur la figure 4. Seuls les pavés pour lesquels il n'a pas été possible de prouver qu'ils ne contenaient aucun minimiseur global sont représentés. Ces résultats ont été obtenus en 3 h sur un Pentium IV à 2 GHz. La symétrie de la projection sur le plan (p_2, p_3) de l'ensemble obtenu suggère un problème d'identifiabilité qui peut être confirmé par une étude théorique (Walter *et al.*, 1997) : les valeurs des paramètres p_2 et p_3 peuvent être échangées sans modification de la sortie du modèle.

7. Conclusions

L'identification est par nature une activité pluridisciplinaire, et les automaticiens identificateurs sont habitués à discuter avec leurs collègues des domaines d'applica-

tion auxquels ils s'intéressent. Les quelques exemples mentionnés ici nous semblent illustrer l'intérêt qu'il y a à regarder aussi du côté des outils méthodologiques développés par d'autres communautés scientifiques. C'est ainsi que l'analyse numérique, les statistiques robustes, la différentiation automatique, le calcul par intervalles et l'intégration garantie nous semblent de nature à modifier profondément notre approche de l'identification des systèmes. Au moins dans le cas de la différentiation automatique, il s'agit d'un juste retour des choses puisque le concept de code adjoint est directement inspiré de travaux de commande optimale déjà anciens.

8. Bibliographie

Berz M., Makino K., « Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models », *Reliable Computing*, vol. 4, n° 4, p. 361-369, 1998.

Bilardello P., Joulia X., Le Lann J. M., Delmas H., Koehret B., « A general strategy for parameter estimation in differential-algebraic systems. », *Computers and Chemical Engineering*, vol. 17, p. 517-525, 1993.

Box G. E. P., Draper N. R., « Robust designs », *Biometrika*, vol. 62, n° 2, p. 347-352, 1975.

Box G. E. P., Hunter J. S., Hunter W. G., *Statistics for Experimenters, 2nd Edition*, Wiley, Hoboken, NJ, 2005.

Farebrother R. W., « The historical developments of the L_1 and L_∞ estimation procedures 1793-1930 », *Statistical Data Analysis Based on the L_1-Norm and Related Methods*, North Holland, Amsterdam, p. 37-63, 1987.

Gear C. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice Hall, Upper Saddle River, N.J., 1971.

Giering R., Kaminski T., « Recipes for adjoint code construction », *ACM Transactions on Mathematical Software (TOMS) archive*, vol. 24, n° 4, p. 437-474, 1998.

Hammer R., Hocks M., Kulisch U., Ratz D., *C++ Toolbox for Verified Computing*, Springer-Verlag, Berlin, 1995.

Hampel F. R., « A general qualitative definition of robustness », *Annals of Mathematical Statistics*, vol. 42, p. 1887-1896, 1971.

Hampel F. R., Ronchetti E. N., Rousseeuw P. J., Stahel W. A., *Robust Statistics. The Approach Based on Influence Functions*, Wiley, New-York, 1986.

Hansen E. R., *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, NY, 1992.

Hassaine Y., Delourme B., Panciatici P., Walter E., « M-Arctan estimator based on the trust-region method », *Electrical Power and Energy Systems*, vol. 28, p. 590-598, 2006.

Huber P., *Robust Statistics*, Wiley, New-York, 1981.

IEEE Computer Society, IEEE Standard for Binary Floating-Point Arithmetic, Technical Report n° IEEE Std. 754-1985, American National Standards Institute, 1985. Available at : http ://standards.ieee.org/.

Jacquez J. A., *Compartmental Analysis in Biology and Medicine*, Elsevier, Amsterdam, 1972.

Jaulin L., Kieffer M., Didrit O., Walter E., *Applied Interval Analysis*, Springer-Verlag, London, 2001.

Jaulin L., Kieffer M., Walter E., Meizel D., « Guaranteed robust nonlinear estimation with application to robot localization », *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 32, n° 4, p. 374-381, 2002.

Kieffer M., Walter E., « Guaranteed parameter estimation for cooperative systems », *in* L. Benvenuti, A. De Santis, L. Farina (eds), *Positive Systems – LNCIS 294*, Springer, Berlin, p. 103-110, 2003.

Lahanier H., Walter E., Gomeni R., « OMNE : a new robust membership-set estimator for the parameters of nonlinear models », *Journal of Pharmacokinetics and Biopharmaceutics*, vol. 15, p. 203-219, 1987.

Lohner R., « Enclosing the solutions of ordinary initial and boundary value problems », *in* E. Kaucher, U. Kulisch, C. Ullrich (eds), *Computer Arithmetic : Scientific Computation and Programming Languages*, BG Teubner, Stuttgart, p. 255-286, 1987.

Merrill H. M., Schweppe F. C., « Bad data suppression in power system static state estimation », *IEEE Trans. on Power Apparatus and Systems*, vol. 90, n° 6, p. 2718-2725, 1971.

Mili L., Cheniae M. G., Vichare N. S., Rousseeuv P. J., « Robust state estimation based on projection statistics », *IEEE Trans. on Power Systems*, vol. 11, n° 2, p. 1118-1127, 1996.

Mili L., Phaniraj V., Rousseeuw P. J., « Robust estimation theory for bad data diagnostics in electrical power systems », *in* C. T. Leondes (ed.), *Control and Dynamic Systems : Advances in Theory and Applications*, Academic Press, San Diego, p. 271-325, 1990.

Müller M., « Über Das Fundamentaltheorem in der Theorie der gewöhnlichen Differentialgleichungen », *Math. Z.*, vol. 26, p. 619-645, 1926.

Naumann U., Utke J., Heimbach P., Hill C., Ozyurt D., Wunsch C., Fagan M., Tallent N., Strout M., Adjoint Code by Source Transformation with OpenAD/F, Technical Report n° AIB-2006-05, RWTH Aachen, May, 2006.

Nedialkov N. S., Jackson K. R., « Methods for initial value problems for ordinary differential equations », *in* U. Kulisch, R. Lohner, A. Facius (eds), *Perspectives on Enclosure Methods*, Springer-Verlag, Vienna, p. 219-264, 2001.

Neumaier A., « Complete search in continuous global optimization and constraint satisfaction », *in* A. Iserles (ed.), *Acta Numerica*, Cambridge University Press, p. 271-369, 2004.

NIST/SEMATECH, *e-Handbook of Statistical Methods*, http ://www.itl.nist.gov/div898/handbook/, 2006.

Pearson E. S., Chandra Sekar C., « The efficiency of statistical tools and a criterion for the rejection of outlying observations », *Geometrika*, vol. 28, p. 308-320, 1936.

Rall L. B., *Automatic Differentiation : Techniques and Applications*, vol. 120 of *LNCIS*, Springer-Verlag, Berlin, Germany, 1981.

Rousseeuw P. J., « Least median of squares regression », *Journal of the American Statistical Association*, vol. 79, n° 388, p. 871-880, 1984.

Rousseeuw P. J., Leroy A. M., *Robust Regression and Outlier Detection*, Wiley, New York, 1987.

Schweppe F. C., Handschin E., « Static state estimation in electric power systems », *Proceedings of the IEEE*, vol. 62, n° 7, p. 972-982, 1974.

Speelpenning B., Compiling Fast Partial Derivatives of Functions Given by Algorithms, Phd dissertation, Dpt of Computer Science, University of Illinois at Urbana-Champaign, 1980.

Talagran O., « The use of adjoint equations in numerical modelling of the atmospheric circulation », *in* A. Griewank, G. Corliss (eds), *Automatic Differentiation of Algorithms : Theory, Implementation, and Application*, SIAM, Philadelphia, p. 169-180, 1991.

Tuckey J. W., « A survey of sampling from contaminated distributions », *in* I. Olkin (ed.), *Contributions to Probability and Statistics*, Stanford, p. 448-485, 1960.

Valko P., Vajda S., « An extended ODE solver for sensitivity calculations », *Computers & Chemistry*, vol. 8, n° 43, p. 255-271, 1984.

Walter E., Kieffer M., « Guaranteed optimisation of the parameters of continuous-time knowledge-based models », *in* C. Commault, N. Marchand (eds), *Positive Systems*, vol. 341 of *LNCIS*, Springer, Heidelberg, p. 137-144, 2006.

Walter E., Pronzato L., *Identification des modèles paramétriques à partir de données expérimentales*, Masson, Paris, 1994.

Walter E., Pronzato L., *Identification of Parametric Models from Experimental Data*, Springer-Verlag, London, 1997.

ANNEXE POUR LE SERVICE FABRICATION

A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
   RS - JESA – 42/2008. Identification des systèmes

2. AUTEURS :
   Eric Walter — Michel Kieffer

3. TITRE DE L'ARTICLE :
   Sur quelques idées fausses
   ayant des conséquences en identification

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Sur quelques idées fausses...

5. DATE DE CETTE VERSION :
   18 mars 2008

6. COORDONNÉES DES AUTEURS :

   – adresse postale :
   Laboratoire des Signaux et Systèmes
   CNRS – SUPELEC – Univ Paris-Sud
   F-91192 Gif-sur-Yvette, France
   Eric.Walter@lss.supelec.fr, Michel.Kieffer@lss.supelec.fr
   – téléphone : 01 69 85 17 11
   – télécopie : 01 69 85 17 65
   – e-mail : eric.walter@lss.supelec.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `article-hermes.cls`,
   version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :
   Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
   `http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com